EECS 482
Introduction to Operating Systems

Winter 2018

Baris Kasikci

(Thanks, Harsha Madhyastha and Jason Flinn for the slides!)

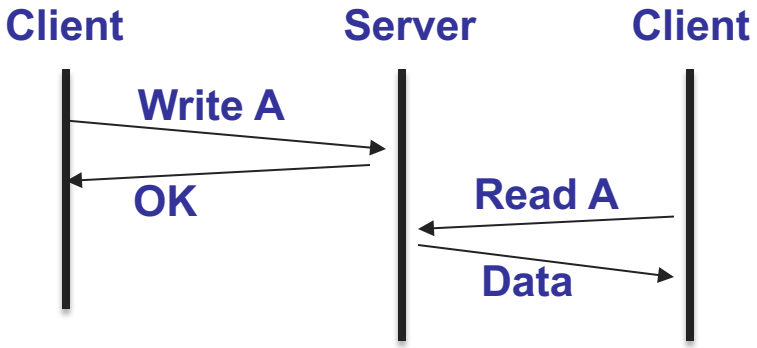
Distributed file systems

- Remote storage of data that appears local
- Examples:
  - AFS
  - Dropbox
  - Google Drive

- Benefits?

Client-server design

- Copy of every file stored on server
- FS operations (read/write/etc.) are RPCs



- Downsides?

Caching for performance

- Can cache file system data at client:
  - In memory (e.g., NFS)
  - On disk (e.g., AFS)

- Benefits of client-side caching:
  - Improves server scalability
  - Better latency and throughput
  - Reduces network traffic
  - Can improve availability (disconnected operation)

Client-side caching

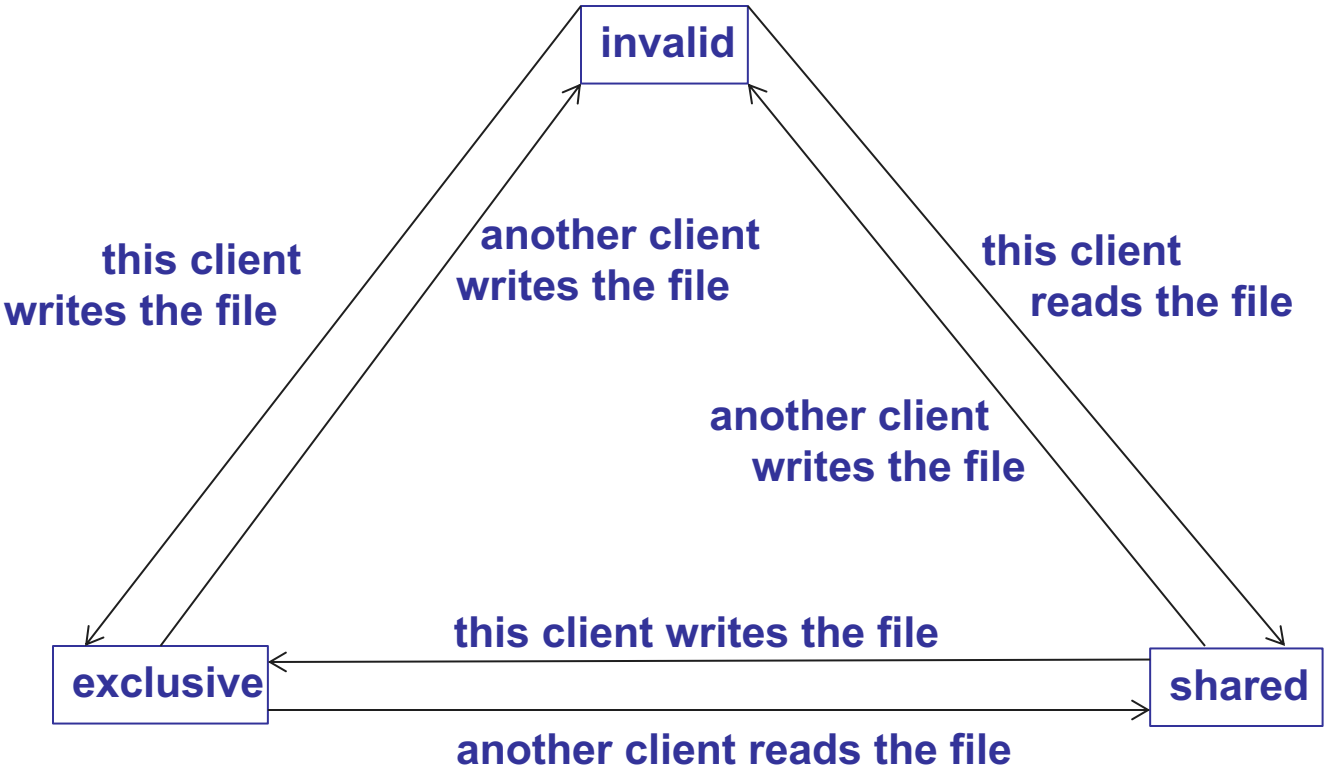
- Migrate or replicate:
  - Migrate: Transfer sole copy from server to client
  - Replicate: Create additional copy at client

Pessimistic concurrency ctrl.

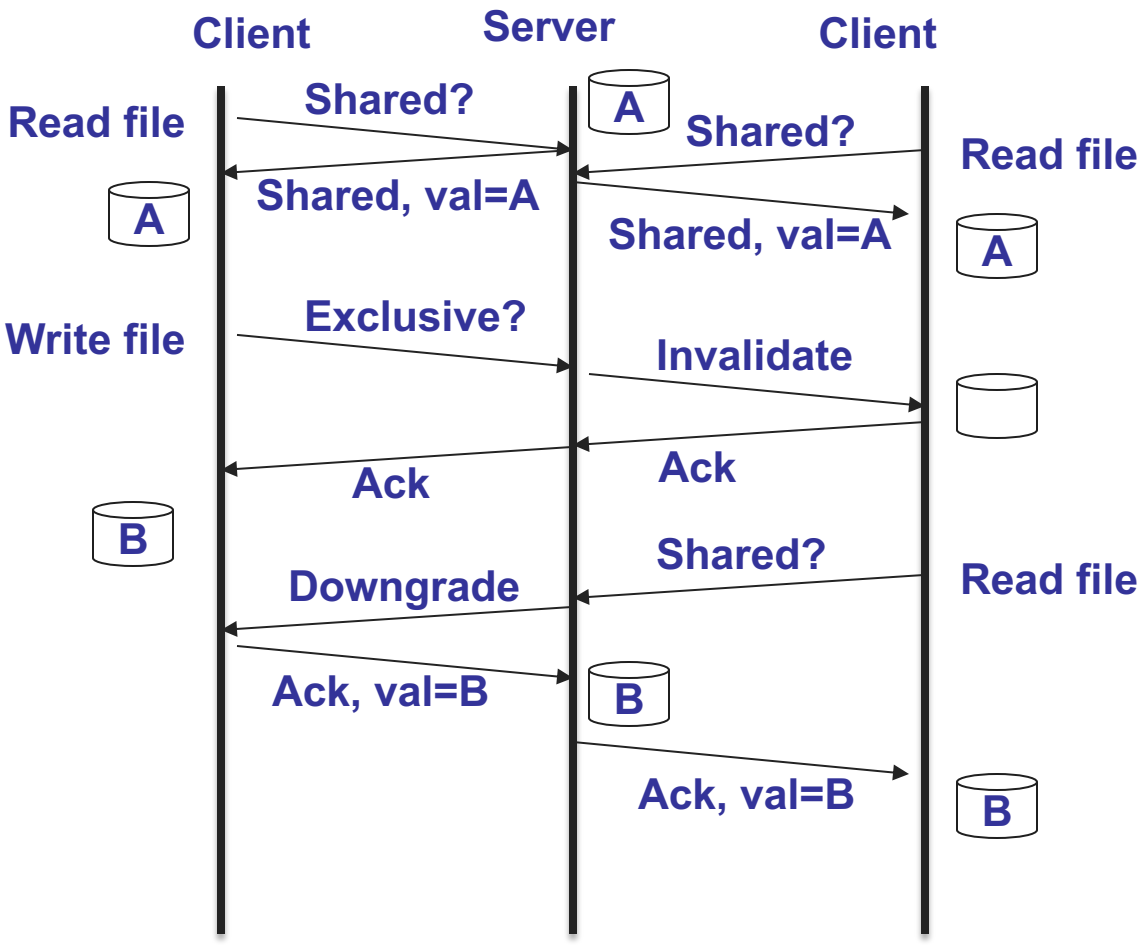
- How to prevent inconsistency?
  - Obtain “lock” before accessing data
  - Similar to reader-writer locks

- 3 states for each replica:
  - Invalid: no cached copy
  - Shared: may read cached copy
  - Exclusive: may read/write cached copy

State machine for cached copy

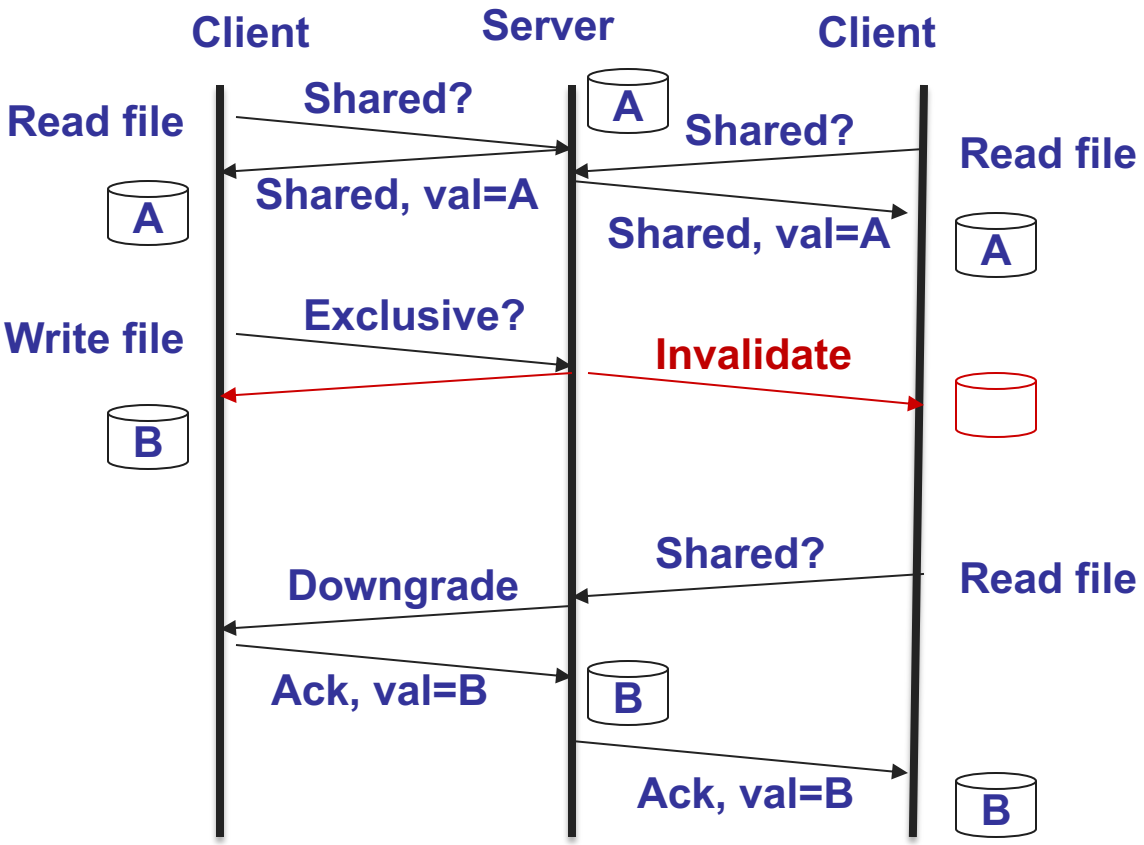


Invalidation protocol



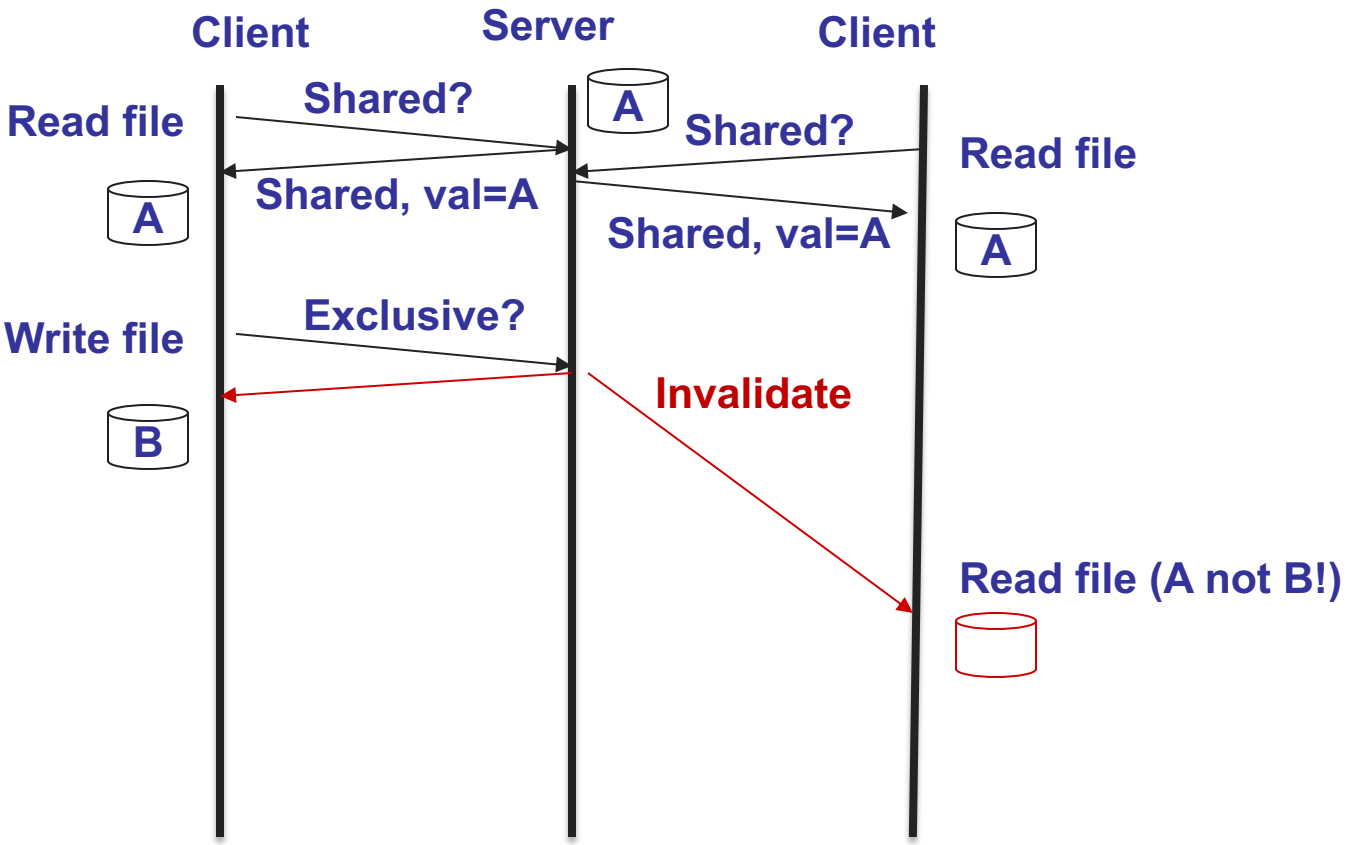
Order of operations

- Necessary to wait for invalidations?



Order of operations

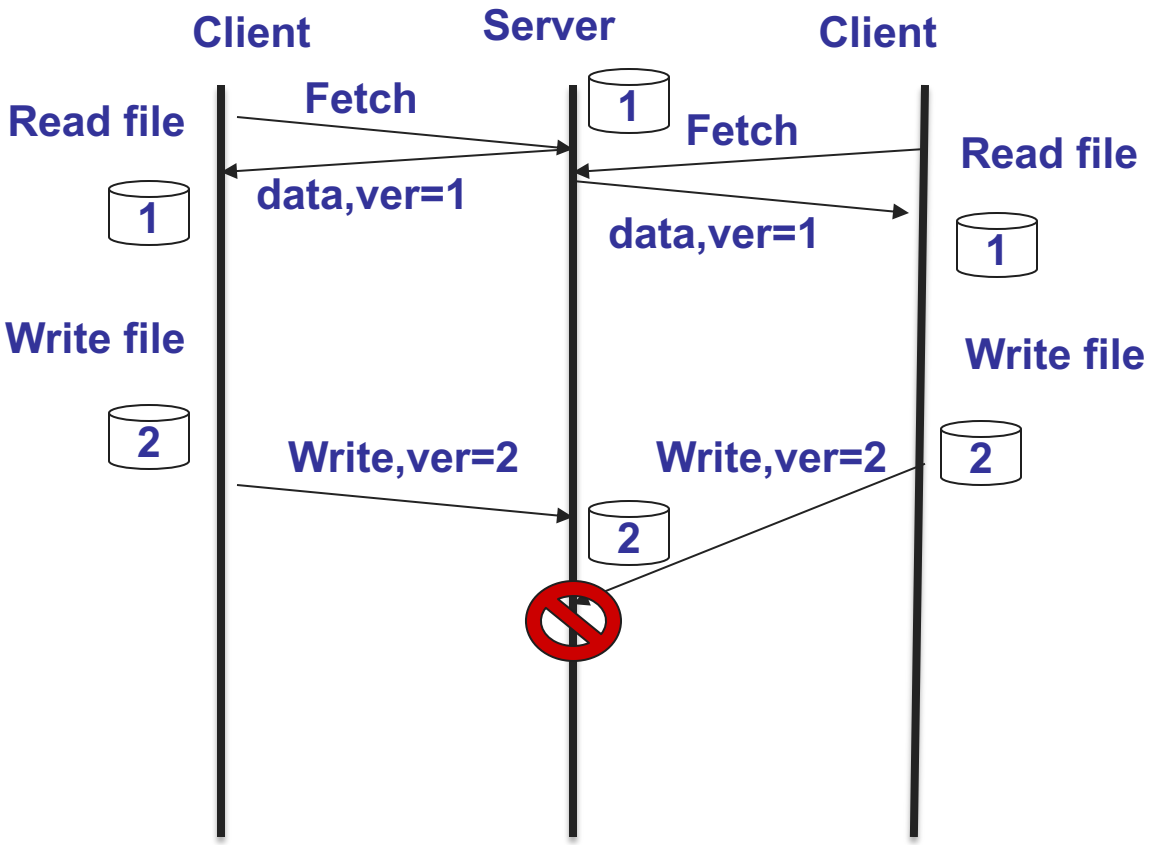
- Necessary to wait for invalidations?



Optimistic concurrency ctrl.

- How to prevent inconsistency?
  - Allow clients to modify replicas freely
  - Server detects and resolves conflicting updates
- How to detect conflicts
  - Assign a version to each object (file, etc.)
  - Increment the version on update
  - Conflict if server version >= client version

Conflicting operations



Resolving conflicts

- Possible strategies?
  - Last writer wins (AFS)
  - Create multiple versions
  - Automatically merge updates (git)
  - Manually merge updates (also git – sigh)
- Jim Gray: no free lunch
  - You either block on a lock or create a conflict
- Are current DFS's optimistic or pessimistic?

Load balancing across servers

- Two options:
  - Partition clients across servers
  - Partition files across servers

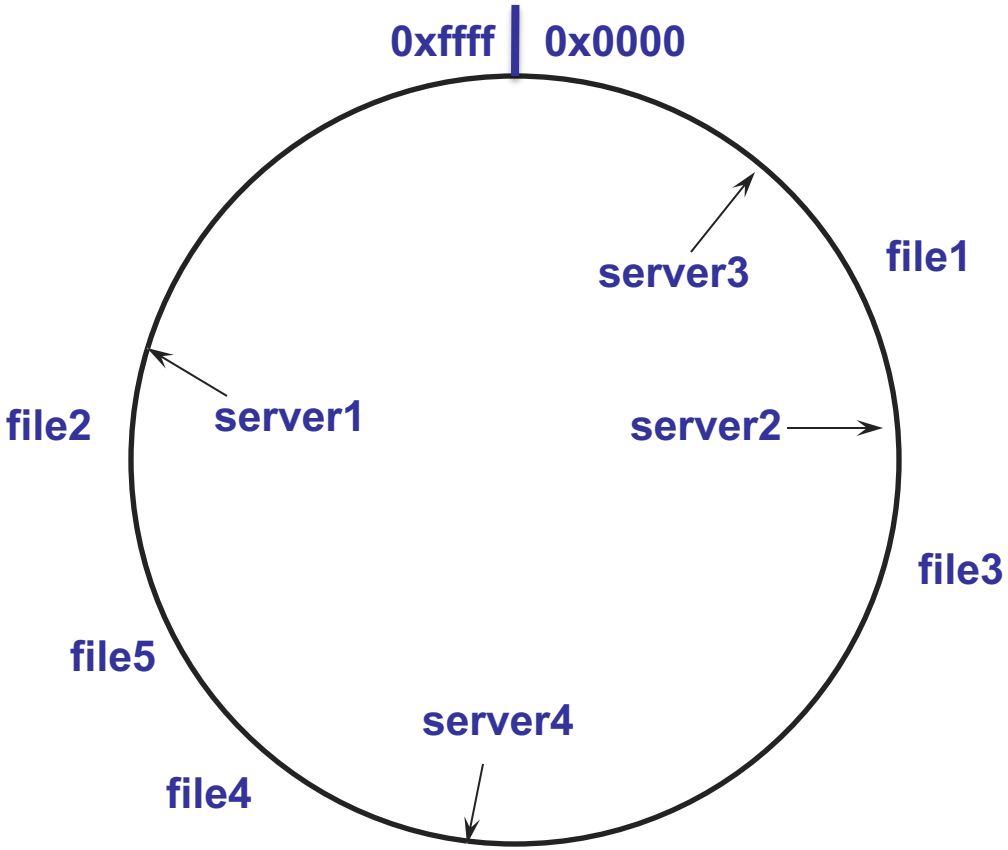
- How to find a file?

Load balancing across servers

A	B	C	D	E
0	1	2	3	
4	5	6	7	
8	9	10	11	

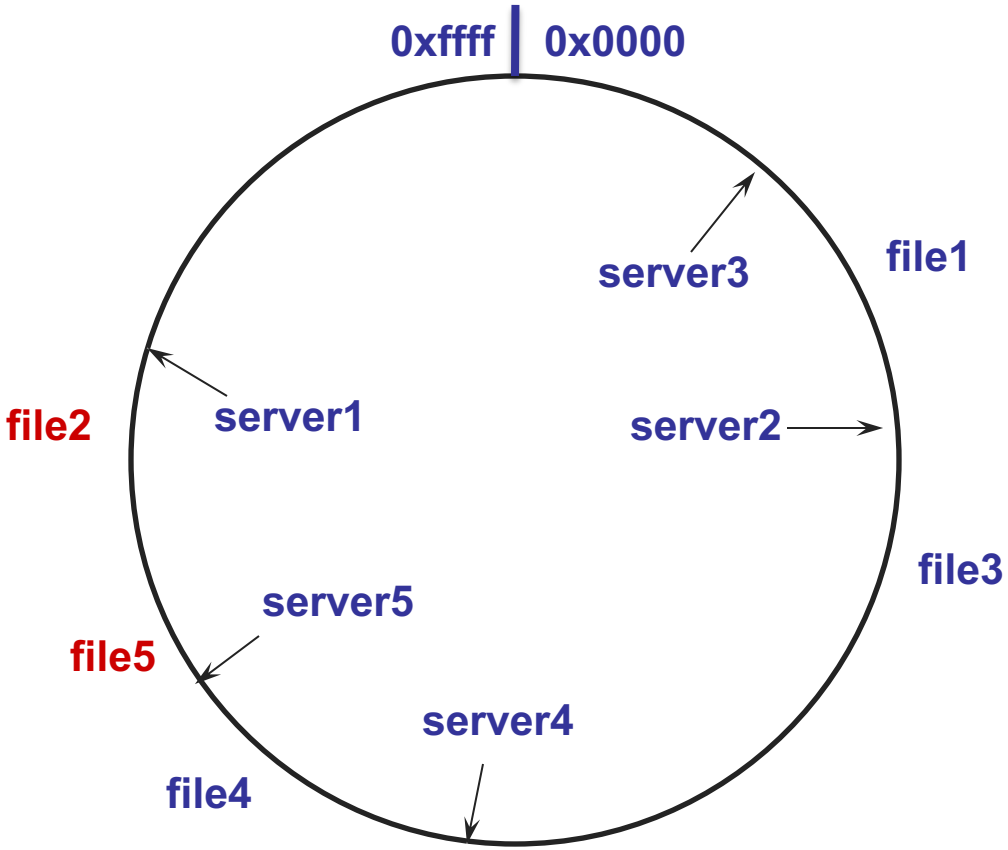
Leveraging randomness

- Hash(server) maps to random location



Adding a server

- Hash(server) maps to random location



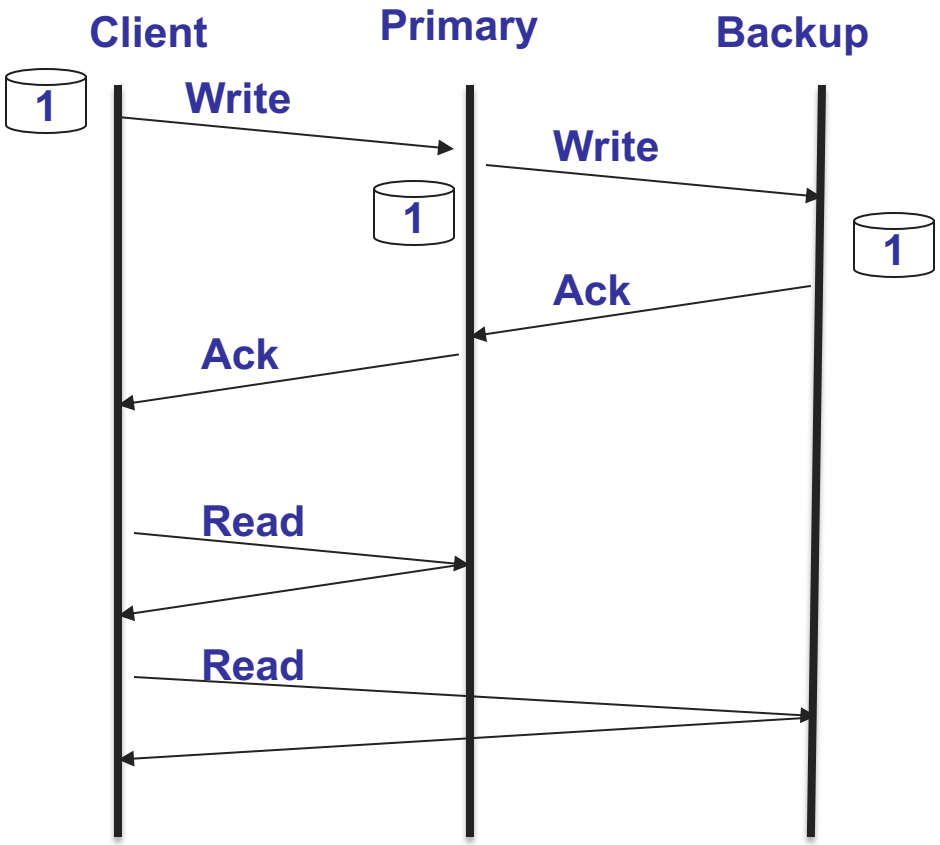
Balancing load

- Some files hotter than others
  - Equal files does not imply equal load
  - How to mitigate?
    - » Assign less regions to hot servers, more to cold

Replication across servers

- More servers → Likelihood of failure increases
- Replicate files to tolerate failures
  - Example: Primary + backup
    - » Write data by writing to both primary AND backup
    - » Read data by reading from primary OR backup

Primary-Backup



Handling failures

- When primary fails
  - Backup becomes new primary
  - Backup handles all reads/writes
  - Primary recovers, syncs state, can become backup
- When backup fails
  - Primary handles all reads/writes
  - Backup recovers, syncs state

Fault tolerance

- What if backup fails before primary recovers?
  - Data is unavailable, lost if failures permanent
- How can we tolerate 2 failures?
  - Use 2 backups
  - Need f+1 servers to tolerate f failures
- What are we assuming about failures?

Byzantine generals

- Say there's one commander C and two lieutenants L1 and L2
  - Goal: Decide whether to attack enemy or retreat
  - Attack succeeds if at least 2 attack
  - 1 of the 3 is a traitor
- Solution 1: L1 and L2 follow C's command to either attack or retreat
- Solution 2: C sends command to L1 and L2, who then exchange notes and follow majority

Byzantine generals

- C sends command to L1 and L2, who then exchange notes and follow majority
- Case 1: L2 is traitor
  - C sends attack to both L1 and L2
  - L1 receives {attack, retreat}
- Case 2: C is traitor
  - C sends attack to L1 and retreat to L2
  - L1 receives {attack, retreat}

Byzantine generals

- Need at least 4 generals to cope with 1 traitor
  - 3f+1 generals to cope with f traitors
  - (more on this in distributed systems classes)
- Solution: C sends command to L1, L2, and L3, who then exchange notes and follow majority
- Three cases:
  - C sends 3 attacks
  - C sends 2 attacks, 1 retreat
  - C sends 1 attack, 2 retreats